A FAST MODE SELECTION ALGORITHM IN H.264 VIDEO CODING

Donghyung Kim and Jechang Jeong

Dept. of Electrical and Computer Engineering
Hanyang University
Haengdang, Seongdong, Seoul, South Korea

ABSTRACT

For improvement of coding efficiency, the H.264 video coding standard uses new coding tools, such as variable block size, quarter-pixel-accuracy motion estimation, multiple reference frames, intra prediction, loop filter, etc. Using these coding tools, H.264 achieves significant improvement in coding efficiency compared with existing standards. However, encoder complexity increases tremendously. Among the tools, the macroblock mode selection and the motion estimation contribute most to total encoder complexity. This paper focuses on complexity reduction in macroblock mode selection. Of all macroblock modes which can be selected, inter8×8 and intra4×4 have the highest complexity. We propose two methods for complexity reduction of inter8×8 and intra4×4 by using the costs of the other macroblock modes. Simulation results show that the proposed methods save up to 57.7% of total encoding time compared with the H.264 reference implementation, whereas the average PSNR only decreases less than 0.05dB.

1. INTRODUCTION

For improvement of coding efficiency, H.264 adopts new coding tools, such as quarter-pixel-accuracy motion estimation (ME), multiple reference frames, loop filter, variable block size (VBS), etc. [1-3]. These tools have enabled the standard to achieve higher coding efficiency than prior video coding standards. The encoder complexity, however, increases tremendously.

Several approaches have been proposed to reduce the H.264 encoder complexity. Yin et al. proposed a method to alleviate the encoder complexity caused by ME and macroblock mode selection [4]. Their low complexity ME algorithm consists of two steps. First, integer-pixel ME is carried out using enhanced prediction zonal search (EPZS). Then, depending on the result of the integer-pixel ME, sub-pixel ME is carried out within some limited areas. For faster macroblock mode selection their method simply examines limited modes based on the costs of inter16×16, inter8×8, and inter4×4. Huang et al. proposed an algorithm to reduce the time to search the reference frames for ME complexity reduction [5]. For each macroblock, they analyze the available information after intra prediction and ME from the previous frame to determine whether it is necessary to search more frames. Their method can save about 10-67% of ME computation. Ahmad et al. proposed a fast algorithm for macroblock mode selection based on a 3D recursive search algorithm that takes into account the cost and the previous frame information [6]. This algorithm leads to over 30% decrease in encoding time compared with the H.264 reference implementation. The bitstream length, however, increases by about 15%.

To speed up the H.264 encoding time, we focus on complexity reduction of the macroblock mode selection. When 8×8 DCT is not used, the candidate macroblock modes are SKIP, inter16×16, inter16×8, inter8×16, inter8×8, intra16×16, and intra4×4. An inter8×8 mode can be further partitioned into four sub-macroblock modes: inter8×8, inter8×4, inter4×8, and inter4×4. Among these modes, inter8×8 and intra4×4 modes contribute most to the complexity, especially when rate-distortion optimization (RDO) is used.

In this paper, we propose two algorithms. One is to alleviate inter8×8 complexity. It estimates four sub-macroblock modes within inter8×8 by using the costs of other inter modes with relatively low complexity. The other method reduces intra4×4 complexity, using similarity between the RD costs of two intra modes.

2. MODE SELECTION ALGORITHM IN THE H.264 REFERENCE SOFTWARE

2.1. Macroblock and Sub-macroblock Modes

The H.264 standard allows the following macroblock modes: SKIP, inter16×16, inter16×8, inter8×16, inter8×8, intra16×16, intra8×8, and intra4×4. Furthermore, each block within inter8x8 can be divided into four sub-macroblock modes. The allowed sub-macroblock modes are inter8×8, inter8×4, inter4×8, and inter4×4. Fig. 1 depicts the macroblock partitions of inter and intra macroblock modes including SKIP mode.

An inter16×16 mode has only one motion vector, whereas inter16×8 and inter8×16 have two motion vectors. An inter8×8 mode may have 4-16 motion vectors depending on the selected sub-macroblock modes. A SKIP mode refers to the mode where neither motion vector nor residual is encoded. Three intra modes have different prediction modes.

This work was supported in part by the Human Resource Development Project for IT SoC Key Architect under Korea IT Industry Promotion Agency.

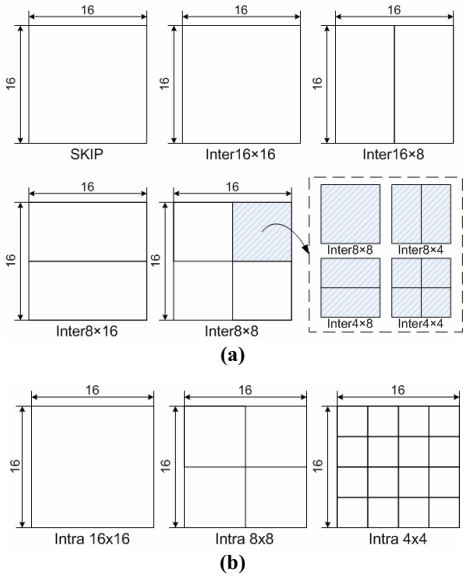

Figure 1. Macroblock partitions of inter(a) and intra(b) modes.

Four prediction modes are available in intra 16x16, and nine prediction modes in intra8x8 and intra4x4.

2.2. Macroblock Mode Selection in Joint Model (JM)

The reference software, JM9.3 [7], supports three cost calculation criteria: motion vector (MV) cost, reference frame (REF) cost, and rate distortion (RD) cost. The MV cost is calculated by using a lambda factor defined as:

$$\begin{aligned} \text{MVcost} &= \textit{WeightedCost(f,mvbits[(cx << s) - px]} \\ &\quad + \textit{mvbits[(cy << s) - py])} \end{aligned} \tag{1}$$

where,

f : *lambda factor*

cx, cy : *candidate x and y position for ME*

px, py : *predicted x and y position for ME*

The REF cost is also calculated by using a lambda factor defined as:

$$\begin{aligned} &\text{REFcost} = \textit{WeightedCost(f,refbits(ref))} \\ &\textit{where, } f \textit{ : lambda factor} \end{aligned} \tag{2}$$

In (1) and (2), *WeightedCost( )* returns the cost for the bits of motion vector and reference frame, respectively. Finally, the RD cost is defined as:

$$\begin{aligned} &\text{RDcost} = \textit{Distortion} + \lambda \cdot \textit{Rate} \\ &\textit{where}, \lambda : \textit{Lagrange multiplier} \end{aligned} \tag{3}$$

In (3), the distortion is computed by calculating the SNR of the block and the rate is calculated by taking into consideration the length of the stream after the last stage of encoding.

When RDO and five reference frames are used, using these cost functions, the process of macroblock mode selection in the reference software is as follows:

Step 1 Find reference frames and motion vectors for each block in inter16×16, inter16×8, and inter8×16.

$$\begin{aligned} &[\mathbf{MV}_i, \mathbf{REF}_i] = \arg\min_{MV,REF} (\text{MVcost}(MV) + \text{REFcost}(REF)) \\ &\textit{where, } i = \textit{inter16} \times \textit{16, inter16} \times \textit{8, inter8} \times \textit{16.} \\ &MV \in \{\textit{Search Range}\},\ REF \in \{0,1,\cdots,4\} \\ &\mathbf{MV}_i : \textit{Motion vector set in i mode} \\ &\mathbf{REF}_i : \textit{Reference frame set in i mode} \end{aligned} \tag{4}$$

Step 2 Calculate the sums of MV cost and REF cost in inter16×16, inter16×8, and inter8×16.

$$\begin{aligned} &J_i = \text{MVcost}(\mathbf{MV}_i) + \text{REFcost}(\mathbf{REF}_i) \\ &\textit{where, } i = \textit{inter16} \times \textit{16, inter16} \times \textit{8, inter8} \times \textit{16.} \\ &J_i : \textit{the sum of } \text{MV cost } \textit{and } \text{REF cost } \textit{in i mode.} \end{aligned} \tag{5}$$

Step 3 Find reference frames and motion vectors for the 1st sub-macroblock in inter8×8.

$$\begin{aligned} &[\mathbf{MV}_i, \mathbf{REF}_i] = \arg\min_{MV,REF} (\text{MVcost}(MV) + \text{REFcost}(REF)) \\ &\textit{where, } i = \textit{inter8} \times \textit{8, inter8} \times \textit{4, inter4} \times \textit{8, inter4} \times \textit{4.} \end{aligned} \tag{6}$$

Step 4 Calculate the sums of MV cost and REF cost for the 1st sub-macroblock in inter8×8.

$$\begin{aligned} &J_i = \text{MVcost}(\mathbf{MV}_i) + \text{REFcost}(\mathbf{REF}_i) \\ &\textit{where, } i = \textit{inter8} \times \textit{8, inter8} \times \textit{4, inter4} \times \textit{8, inter4} \times \textit{4.} \end{aligned} \tag{7}$$

Step 5 Select the mode for the 1st sub-macroblock in inter8×8.

$$\begin{aligned} &\textit{Sub - macroblock mode} = \\ &\arg\min_{mode} (\text{RDcost}(\textit{inter8} \times \textit{8}), \text{RDcost}(\textit{inter8} \times \textit{4}) \\ &\quad , \text{RDcost}(\textit{inter4} \times \textit{8}), \text{RDcost}(\textit{inter4} \times \textit{4})) \end{aligned} \tag{8}$$

Step 6 Repeat Steps 3 to 5 for the other sub-macroblocks in inter8×8.

Step 7 Select the macroblock mode

$$\begin{aligned} &\textit{Macroblock mode} = \\ &\arg\min_{mode} (\text{RDcost}(\textit{SKIP}), \text{RDcost}(\textit{inter16} \times \textit{16}) \\ &, \text{RDcost}(\textit{inter16} \times \textit{8}), \text{RDcost}(\textit{inter8} \times \textit{16}), \text{RDcost}(\textit{inter8} \times \textit{8}) \\ &\quad , \text{RDcost}(\textit{intra16} \times \textit{16}), \text{RDcost}(\textit{intra4} \times \textit{4})) \end{aligned} \tag{9}$$

In Step 1 and Step 2, the reference software finds reference frames and motion vectors which minimize the sum of MV cost and REF cost in inter16×16, inter16×8, and inter8×16. Steps 3 to 6 are the process to select sub-macroblock modes in inter8×8. The final step decides the macroblock mode by comparing RD costs of all macroblock modes.

3. PROPOSED ALGORITHM

3.1. Complexity Reduction of Inter8×8

Since each sub-macroblock within inter8×8, for the selection of sub-macroblock modes, needs additional RD cost computations, inter8×8 has the highest complexity among all inter macroblock modes. For complexity

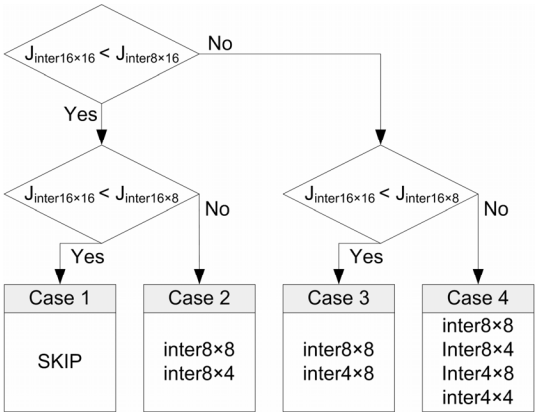

Figure 2. Restriction of selectable sub-macroblock modes.

reduction of inter8×8 we assume that the costs of inter macroblock modes monotonically increase or decrease according to partitioned direction. Using this assumption, we restrict selectable sub-macroblock modes by using the MV costs and REF costs of inter16×16, inter16×8, and inter8×16. For example, if the sum of MV and REF costs of inter16×16 is larger than that of inter16×8 and is smaller than that of inter8×16, we only consider inter8×8 and 8×4 as sub-macroblock modes. Fig. 2 depicts the proposed method for the complexity reduction of inter8×8.

In case 1, since $J_{inter16\times16}$ is smaller than both $J_{inter16\times8}$ and $J_{inter8\times16}$, neither additional block partition in horizontal direction nor in vertical direction is needed. In this case, we do not consider any sub-macroblock mode, and the process of Step 3 to Step 6 in the reference software is skipped. In case 2, since $J_{inter16\times16}$ is smaller than $J_{inter8\times16}$ and is larger than $J_{inter16\times8}$, additional block partition is only considered in the vertical direction. In this case, either inter8×8 and inter8×4 is selected as a sub-macroblock mode, and the formulae of Steps 3 to 6 in the reference software are modified as follows:

$$[\mathbf{MV}_i, \mathbf{REF}_i] = \arg\min_{MV,REF} (\text{MVcost}(MV) + \text{REFcost}(REF)) \quad (10)$$

where, i = inter8×8, inter4×8.

$$J_i = \text{MVcost}(\mathbf{MV}_i) + \text{REFcost}(\mathbf{REF}_i) \quad (11)$$

where, i = inter8×8, inter8×4.

$$Sub-macroblock\ mode = \arg\min_{mode} (\text{RDcost}(inter8\times8), \text{RDcost}(inter8\times4)) \quad (12)$$

In case 3, since $J_{inter16\times16}$ is smaller than $J_{inter16\times8}$ and is larger than $J_{inter8\times16}$, only additional block partition only in the horizontal direction is considered. In this case, either inter8×8 and inter4×8 is selected as a sub-macroblock mode, and the formulae of Steps 3 to 6 in the reference software are modified as follows:

$$[\mathbf{MV}_i, \mathbf{REF}_i] = \arg\min_{MV,REF} (\text{MVcost}(MV) + \text{REFcost}(REF)) \quad (13)$$

where, i = inter8×8, inter4×8.

$$J_i = \text{MVcost}(\mathbf{MV}_i) + \text{REFcost}(\mathbf{REF}_i) \quad (14)$$

where, i = inter8×8, inter4×8.

$$Sub-macroblock\ mode = \arg\min_{mode} (\text{RDcost}(inter8\times8), \text{RDcost}(inter4\times8)) \quad (15)$$

In case 4, since $J_{inter16\times16}$ is larger than both $J_{inter16\times8}$ and $J_{inter8\times16}$, we consider all sub-macroblock modes, as in the reference software.

3.2. Complexity Reduction of Intra4×4

When 8×8 DCT is not used, the allowed intra modes are intra4×4 and intra16×16. Of the two intra modes, intra4×4 has the higher complexity because it has more prediction modes. Since intra16×16, as described in Section 2, has only four prediction modes and intra4×4 has nine prediction modes for finer size, intra4×4 generally yields smaller prediction error than intra16×16.

However, most of the macroblocks have only small difference between the RD costs of intra16x16 and intra4x4. It is because edges directed in vertical or horizontal are dominant in natural images, which are considered in intra16x16.

Using this characteristic, we first find the inter mode with a minimum RD cost. Then we compare the RD cost of the selected inter mode with that of intra16×16. If the RD cost of intra16×16 is much larger, that is, Eq. (16) is satisfied, than the RD cost computation of intra4×4 is skipped:

$$\text{Min}[\text{RDcost}(inter\ modes)] \cdot K < \text{RDcost}(intra\ 16\times16) \quad (16)$$

In (16), K is a constant. Table 1 describes the missing rate of intra4×4. The missing rate indicates the probability that the skipped intra4×4 has the smallest RD cost. As shown in Table 1, the average missing rate is only about 0.7% for K =1.5. This means that the RD cost difference between intra4×4 and intra16×16 is less than 1.5 times for 99.3% of the macroblocks.

4. SIMULATION RESULTS

Since the proposed methods for complexity reduction of inter8×8 and intra4×4 are uncorrelated, the two methods can be applied independently, or simultaneously. We applied the two proposed algorithms simultaneously to encode test sequences. For the purpose of evaluation, the public reference encoder JVT Model (JM) v.9.3 was used. The software was tested on an Intel Pentium-IV based computer with 512 MB RAM under the Windows XP Professional operating system.

TABLE 1. MISSING RATE OF INTRA4×4 ACCORDING TO K

Sequences	Missing Rate (%)		
	K =1.3	K =1.5	K =1.7
Coastguard	2.0	0.4	0.2
Container	2.7	1.0	0.9
Mobile	0.4	0.0	0.0
News	5.8	0.6	0.5
Salesman	6.6	0.4	0.0
Silent	4.7	1.7	0.4
Stefan	2.8	0.2	0.0
Trevor	9.0	0.9	0.1

TABLE 2. THE NUMBER OF RD COST COMPUTATIONS IN INTER8×8.

Sequences	Reference Software	Proposed Method	Reduction Ratio (%)
Coastguard	156,816	59,760	62
Container	156,816	17,896	89
Mobile	156,816	65,360	58
News	156,816	30,256	81
Salesman	156,816	28,224	82
Silent	156,816	40,464	74
Stefan	156,816	56,472	64
Trevor	156,816	54,800	65

TABLE 3. THE NUMBER OF RD COST COMPUTATIONS IN INTRA4×4.

Sequences	Reference Software	Proposed Method	Reduction Ratio (%)
Coastguard	35,343	10,838	69.3
Container	35,343	6,566	81.4
Mobile	35,343	400	98.9
News	35,343	2,225	93.7
Salesman	35,343	529	98.5
Silent	35,343	2,388	93.2
Stefan	35,343	3,038	91.4
Trevor	35,343	3,463	90.2

TABLE 4. COMPARISON OF BITRATES (Kbits).

Sequences	Reference Software	Proposed Method	Increase Ratio (%)
Coastguard	249.00	251.28	0.9
Container	40.16	40.74	1.4
Mobile	496.49	497.24	0.2
News	75.84	76.75	1.2
Salesman	56.89	57.61	1.3
Silent	82.69	83.71	1.2
Stefan	379.26	380.86	0.4
Trevor	132.49	133.58	0.8

TABLE 5. COMPARISON OF PSNRS.

Sequences	Reference Software (dB)	Proposed Method (dB)	Difference (dB)
Coastguard	33.93	33.89	0.04
Container	36.07	36.06	0.01
Mobile	33.14	33.06	0.08
News	36.65	36.64	0.01
Salesman	35.57	35.54	0.03
Silent	35.84	35.81	0.03
Stefan	34.22	34.15	0.07
Trevor	36.40	36.33	0.07

We adopted full search for ME, used RDO, and set Quantization Parameter (QP) and K in (16) to 28 and 1.5, respectively. The simulation was performed on eight standard video sequences in QCIF (176x144) format. These included Coastguard, Container, Mobile, News, Salesman, Silent, Stefan, and Trevor. These sequences were selected on the basis of length of encoded streams and degree of motion. The first 100 frames of each of these sequences were used.

For 99 frames, Tables 2 and 3 describe the reduction ratios of the number of RD cost computations in inter8x8 and intra4x4. As shown in these results, we can save about 72%, 90% of the RD cost computations, respectively.

Tables 4 and 5 compare the bitrates and PSNRs for each test sequence. Since the reference implementation is an exhaustive search for selecting the macroblock mode, the number of encoded bits is the least for each sequence. Tables 4 and 5 show the average increase of the total bitrates is only about 0.9%, and the average PSNR drop is only about 0.043dB when using the proposed method.

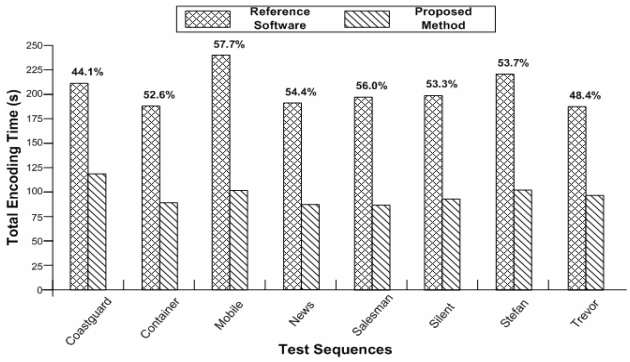

Figure 3. Comparison of total encoding time

Finally, Fig. 3 compares total encoding time from the proposed method with that from the reference software. This result shows substantial decrease of about 53% in total encoding time compared with the reference implementation.

5. CONCLUSION

We proposed two simple and effective schemes for quick selection of macroblock modes in H.264 video coding. Using our methods, the RD cost computations of inter8×8 and intra4×4 was reduced about 72% and 90%, respectively. Both schemes can be applied independently. When both methods are used simultaneously, simulation results show that our methods can save about 53% of total encoding time regardless of input sequences, yet the average increased rate of the total bits and average PSNR drop are only about 0.9% and 0.043dB, respectively. This huge reduction of encoder complexity may be useful in real-time implementation of the H.264/AVC standard.

7. REFERENCES

[1] JVT K051, "Version 3 of H.264/AVC (ITU-T Rec. H.264/ISO/IEC 14496-10 AVC)," July 2004.

[2] T. Wiegand and G. J. Sullivan, "Overview of the H.264/AVC Video Coding Standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, pp. 560-576, July 2003.

[3] T. Wiegand, H. Schwarz, A. Joch, and F. Kossentini, "Rate-Constrained Coder Control and Comparison of Video Coding Standard," *IEEE Trans. Circuits Syst. Video Technol*., vol. 13, pp. 688-703, July 2003.

[4] P. Yin, H. C. Tourapis, A. M. Tourapis, and J. Boyce, "Fast Mode Decision and Motion Estimation for JVT/H.264," *ICIP'03*, vol. 3, pp. 853-856, Sept. 2003.

[5] Y. W. Huang, B. Y. Hsieh, T. C. Whang, S. Y. Chien, S. Y. Ma, C. F. Shen, and L. G. Chen, "Analysis and Reduction of Reference Frames for Motion Estimation in MPEG-4 AVC/JVT/H.264," *ICASSP'03*, vol. 3 pp. 145-148, April 2003.

[6] A. Ahmad, N. Khan, S. Masud, and M.A. Maud, "Efficient block size selection in H.264 video coding standard," *Electronics Letters*, vol. 40, pp. 19-21, Jan. 2004.

[7] JM9.3: http://bs.hhi.de/~suehring/tml/ download/ jm93.zip.